# Indore 10 EAP3 (build 41843) Release Notes

See also Indore 10 EAP2 (build 41463) Release Notes



## Project Settings Configuration Changes

Starting from this EAP, configuration of issue trackers, versioned settings, custom charts, shared resources and third-party report tabs was moved from `<TeamCity Data Directory>/config/projects/<ProjectID>/pluginData/plugin-settings.xml` to `<TeamCity Data Directory>/config/projects/<ProjectID>/project-config.xml` file. The file now has the `<project-extensions>` element which contains all of the above-mentioned project features.

## Two-node Server Configuration

The TeamCity server can operate in two modes now: main TeamCity server (default mode) and build messages processor.

If a TeamCity server is started in the build messages processor mode, its functionality is reduced to only one feature - process real time data coming from the running builds. The primary purpose of this mode is to provide the scalability option for really large installations running several hundreds of agents on a single server now and planning on expanding their installation in the future.

Several important notes about two-node configuration:

- the server started in the build messages processor mode handles all of the data produced by running builds (build logs, artifacts, statistic values), pre-processes it and stores it in the database and on the disk
- in a two-node installation the main TeamCity server handles all other tasks: user interface, VCS-related activity, management of agents, etc.
- both the main TeamCity server and the build messages processor require access to the same data directory (which must be shared if the nodes are installed on separate machines) and to the same database
- the URL where the build messages processor operates must be accessible by the agents and the main TeamCity server (occasionally the main TeamCity server also communicates with the build messages processor by HTTP)

### Installation

First of all, ensure that both machines where TeamCity software will be installed can access the same TeamCity data directory in read/write mode. For large installations we recommend using NAS. It is possible to share the directory using NFS, or Windows share, although the performance of such installation can be worse.

Once you have two machines, proceed with installing TeamCity software as usual: download a distribution package, unpack it or follow the installation wizard.

Configure the TEAMCITY_DATA_PATH environment variable on both machines, make sure it points to the shared data directory.

On the build messages processor machine, add additional arguments to the TEAMCITY_SERVER_OPTS environment variable:

export TEAMCITY_SERVER_OPTS=-Dteamcity.server.mode=build-messages-processor -Dteamcity.server.rootURL=<processor url> <your regular options if you have them>,

where `<processor url>` is the URL of the build messages processor. This URL must be accessible by both the agents and main server. If you do not have an HTTP proxy installed in front of the TeamCity servlet container and you did not change the port in the servlet container during the installation, then by default this URL will be: `http://<your host name>:8111/`

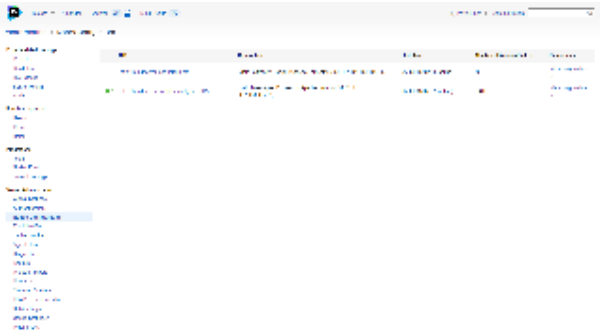
## Startup & shutdown

Both the main TeamCity server and build messages processor can be started / stopped using regular TeamCity scripts (`teamcity-server.bat, teamcity-server.sh`) or Windows services.

The build messages processor uses the same approach to logging as the main server. You can check how the startup is doing in the <TeamCity installation directory>/logs/teamcity-server.log file. You can also open <processor URL> in your browser, there you should see regular TeamCity startup screens.

## Enabling Build Messages Processor

If both the main server and build messages processor are up and running, you should see the Administration | Nodes configuration page on the main TeamCity server:



By default, the build messages processor is disabled and all traffic produced by running builds is still handled by the main server. Once you enable the build messages processor, all newly started builds will be routed to this node. The existing running builds will continue being executed on the main server.

And vice versa, if you decide to disable the messages processor, only the newly started builds will be switched to the main server, the builds that were already running on the messages processor will continue running there.

On this page you can also see how many builds are currently assigned to each node. If the build messages processor is enabled, eventually all of the running builds will be switched to this node.

## Restarting Servers while Builds are Running

The build messages processor as well as the main TeamCity server can be stopped or restarted while builds are running there. If agents cannot connect to the build messages processor for some time, they will re-route their data to the main server. If the main server is unavailable either, agents will keep their data and re-send it once servers re-appear.

## Upgrade & Downgrade

Both the main TeamCity server and the build messages processor must be of exactly the same version.

To upgrade:

1. stop the build messages processor (if you forget to do that, you'll be warned during upgrade)
2. start upgrade on the main TeamCity server as usual
3. proceed with upgrade
4. check everything is ok, agents are connecting etc. (since the messages processor is not available anymore, the agents will re-route their data to the main server)
5. upgrade software on the messages processor machine to the same version
6. start the messages processor and check that it is connected using the Nodes configuration page on the main server

To downgrade:

1. shutdown the main server and build messages processor
2. restore data from backup (only if the data format has been changed during upgrade)
3. downgrade TeamCity software on the main server
4. start the main TeamCity server and check that everything works
5. downgrade TeamCity software on the build messages processor to the same version as the main server
6. start the build messages processor

As usually with TeamCity, agents perform upgrade / downgrade automatically.

### Cleanup

The TeamCity cleanup task runs on the main TeamCity server only. In a two-node configuration as well as in a single node configuration, it can work at the same time while the servers are handling data from the running builds.

### Backup & Restore

Backup through TeamCity web interface can be done on the main TeamCity server only. Backup from the command line can be done on either the build messages processor or on the main TeamCity server.

Restore can be done on either of these nodes, but only if all of the servers using the TeamCity database and data directory are stopped.

### Current limitations

- Only one build messages processor can be configured for the main TeamCity server.
- At the moment the build messages processor node does not support plugins.

## Team Foundation Work Items Tracking

Since TeamCity 10, Team Foundation Work Items tracking is integrated with TeamCity. Supported versions are Microsoft Visual Studio Team Foundation Server 2010-2015, and Visual Studio Team Services.

TFS work items support can be configured on the Issue trackers page for a project. If a project has a TFVC root configured, TeamCity will suggest configuring the issue tracker as well.
By default, the integration works the same way as the other issue tracker integrations: you need to mention the work item ID in the comment message, so the work items can be linked to builds and the links will be displayed in various places in the TeamCity Web UI. Additionally, if your changeset has related work items,TeamCity can retrieve information about them even if no comment is added to the changeset. Besides, custom states for resolved work items are supported by TeamCity.

In addition, resolved states in TeamCity can be customized by using the `teamcity.tfs.workItems.resolvedStates` internal property set to `"Closed?|Done|Fixed|Resolved?|Removed?"` by default.

## Cloud Support

- When configuring a cloud image, you can now select an agent pool for the newly created cloud agents. Previously all of them were placed in the default pool.

### VMware vSphere

- Custom names for agent images are supported now. The names of virtual machines in VMware must be unique. When using the same image in different cloud profiles, to avoid possible conflicts, use the custom agent image name when configuring a cloud profile in TeamCity. This feature can be also useful with naming patterns for agents. When a custom agent image name is specified, the names of cloud agent instances cloned from the image will be based on this name.

### Amazon EC2

- Custom tags can now be applied to EC2 cloud agent instances: when configuring Cloud profile settings, in the Add Image/ Edit Image dialog use the Instance tags: field to specify tags in the format of`<key1>=<value1>,<key2>=<value 2>`. Amazon tag restrictions need to be considered.
- EBS optimization is turned on by default for all instance types supporting it.

## Email Verification

TeamCity administrators can enable / disable email verification (off by default) on the Administration | Authentication page.

If email verification is enabled on the TeamCity server, the Email address field in the user account registration form becomes mandatory. When an address is added / modified, the users will be asked to verify it. If the email address is not verified, TeamCity will display a notification on the General tab of the user account settings. Verified email addresses will be marked with a green check on the Administration | Users page.

During the project import TeamCity will take verified emails into account. If there are two users with different usernames, but the same verified email, TeamCity will provide a possibility to merge these users.

## Exclude Patterns for Artifact Paths

It is now possible to specify newline- or comma-separated paths in the form of `-:source [ => target]` to exclude files or directories from publishing as build artifacts.

Rules are grouped by the right part and are applied in the order of appearance, e.g.

```
+:**/* => target_directory
-:**/folder1 => target_directory
```

will tell TeamCity to publish all files except for `folder1` into the `target_directory`.

## Project-bound Agent Management Permissions

Earlier granting agent-related permissions, like 'authorize' or 'enable' could be done only globally (permissions were applied to all the agents on the server).

Now we introduce project-level permissions to perform a task on an agent: a user must have a specific permission granted in a project.

A user can perform a task controlled by one of these permissions on all the agents belonging to some pool provided this permission is granted to the user in all the projects associated with this pool. For example, a user with 'Enable / disable agents associated with project' permission in some projects can enable or disable agents which belong to the pools of the related projects if the permission is granted in all the projects associated with the pools.

The newly added permissions are:

- Enable / disable agents associated with project
- Start / Stop cloud agent for project
- Change agent run configuration policy for project
- Administer project agent machines (e.g. reboot, view agent logs, etc.)
- Remove project agent
- Authorize project agent

The last permission and the recently introduced "maximum number of agents" setting for an agent pool enable you to set up the system in a way which allows project administrators to run new agents and authorize/add them to their pools without involving the global server administrator.

In new installations, these project-related permissions are added to the Project Administrator role. After upgrade, no permissions changes are applied, but you might want to review the permissions, e.g. consider removing the Agent Manager role inclusion from the Project Administrator role and adding the above-mentioned permissions to Project Administrator role manually.

## Administration | Tools page improvements

In previous versions of TeamCity you could install a couple of predefined tools to TeamCity agents via the Administration | Tools page. In addition to that you could manually distribute zip archives to agents if you place them into special place on disk.

In addition, some of the plugins, like NuGet had own place for uploading new versions to agents.

In this EAP we unified all these tasks under the single interface. Now, you can manage NuGet versions, install recent version of ReSharper command line tools, or upload an arbitrary tool in a zip archive - all from the same Administration | Tools page.

# Flaky Test Detection

TeamCity now supports flaky test detection. A flaky test is a test that is unstable (can exhibit both a passing and a failing result) with the same code.

Flaky test detection is based on the following heuristics:

1. High flip rate (Frequent test status changes). A flip in TeamCity is a test status change — either from OK to Failure, or vice versa. The Flip Rate is the ratio of such "flips" to the invocation count of a given test, measured per agent, per build configuration, or over a certain time period (7 days by default). A test which constantly fails, while having a 100% failure rate, will have its flip rate close to zero; a test which "flips" each time it is invoked will have the flip rate close to 100%.
   If the flip rate is too high, TeamCity will consider the test flaky.
2. Different test status for build configurations with the same VCS change: if two builds of different configurations are run on the same VCS change and the test result is different in these builds, the test is reported as flaky. This may be an indication of environmental issues.
3. If the status of a test 'flipped' in the new build with no changes, i.e. a previously successful test failed in a build without changes or a previously failing test passed in a build without changes, TeamCity will consider the test flaky.
4. Different test status for multiple invocations in the same build: if the same test is invoked multiple times and the test status flips, TeamCity will consider the test flaky.

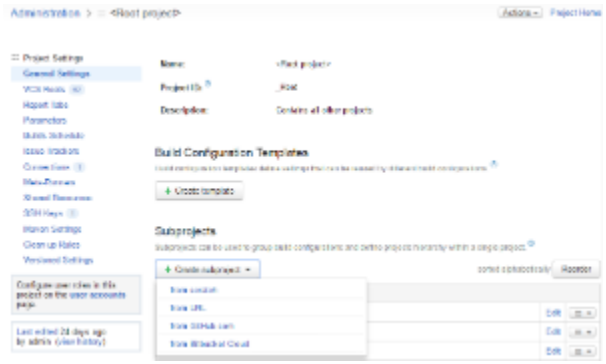
Such tests are displayed on the dedicated project tab, Flaky Tests, along with the total number of test failures, the flip rate for the given test and reasons for qualifying the test as a flaky one. You can also see if the test is flaky when viewing the expanded stacktrace for a failed test on the build results page.

As with any failed test, you can assign investigations for a flaky test (or multiple tests). For flaky tests the resolution method is automatically set to 'Manual'; otherwise the investigation will be automatically removed once the test is successful, which does not mean that the flaky test has been fixed.

Note that is branches are configured for a VCS Root, flaky tests are detected for the default branch only.

# New Create project / Create build configuration buttons

The new Create subproject and Create build configuration buttons have a drop-down now and you can select whether you want to create a project from scratch (manually), from URL, or using the popular version control systems GitHub.com and Bitbucket. When one of the latter two is selected, TeamCity offers to configure a connection to the VCS hosting for the current project. When the connection is configured, TeamCity displays the list of the available repositories with their URLs. All you have to do is select a repository URL and proceed with the configuration.




# REST API Enhancements

- It is possible now to get help on locator usage by sending a request with "$help" string as a locator
- Project features are now exposed
- Order of projects and build configurations can now be changed via .../app/rest/projects/xxx/order/projects and .../app/rest/projects/xxx/order/buildTypes requests
- It is now possible to get and update TeamCity license keys
- Maximum number of agents in an agent pool is supported

# DSL Changes

A dedicated DSL was added for some settings, e.g. for git VCS roots:

```
object Project_Sources : KVcsRoot({
   uuid = "uuid"
   extId = "Project_Sources"
   name = "Sources"

   param("url", "http://acme.com/repo.git")
   param("branch", "refs/heads/master")
   param("teamcity:branchSpec", "+:refs/heads/*")
   param("usernameStyle", "USERID")
   param("reportTagRevisions", "true")
})
```

becomes

```
object Project_Sources : GitVcsRoot({
   uuid = "uuid"
   extId = "Project_Sources"
   name = "Sources"

   url = "http://acme.com/repo.git"
   branch = "refs/heads/master"
   branchSpec = "+:refs/heads/*"
   userNameStyle = UserNameStyle.USERID
   useTagsAsBranches = true
})
```

A similar DSL is provided for other plugins: mercurial, command-line, maven, gradle and others.

'K' prefix is dropped from class names, so KProject becomes Project in DSL.

## Other Improvements

- Maven-related operations performed on the server-side are now moved to separate process
- New option added to Subversion VCS root: Enable non-trusted SSL certificate; if this option is enabled, TeamCity will be able to connect to SVN servers without properly signed SSL certificate
- Starting from this EAP TeamCity uses unidirectional agent-to-server connection via the polling protocol by default. If for some reason the polling protocol cannot be used, TeamCity switches to the fallback bidirectional communication via xml -rpc.
- the bundled IntelliJ IDEA is updated to 2016.2 RC (162.1121.10)
- fixed issues